ISTANBUL AYDIN UNIVERSITY

Project Report

On

ATM Banking System

Submitted By: Hamza Ayman Kosa and Hasan Mahmood Njimuldeen Al-Ani

In

ENGENEERING SCHOOL OF AYDIN UNIVERSITY OF SOFTWARE ENGENEERING

MAY 2020

-

<ATM Banking System>
project Document

-

Authors:

Hamza Ayman Kosa

Student Number,

B1905.090020

&

Hasan Mahmood Njimuldeen Al-Ani

Student Number,

B1905.090005

Document Revision History

Date	Version	Description	Author
12/5/2020	0.0	Ioperations Interface Class, LoginSysten Class, Main Class, MainController Class, MainLoginPage Scene (fxml), User Class	Hamza A. Kosa & Hasan M. Al-Ani
17/5/2020	0.1	UserMainPage Scene (fxml), UserLoginController Class, AdminLoginController Class	Hamza A. Kosa & Hasan M. Al-Ani
24/5/2020	0.2	UserCheckBalance Scene (fxml), UserDepositmoney Scene(fxml), UserTransferMoney Scene (fxml), UserWithdrawPage Scene (fxml), AdminAddUsers Scene (fxml), AdminPageController Class, UserPageController Class	Hamza A. Kosa & Hasan M. Al-Ani
25/5/2020	0.3	AdminAccess Class	Hasan M. Al-Ani
26/5/2020	1.0	AdminMainPage Scene (fxml), AdminChangePassword Scene (fxml), AdminRemoveUsers Scene (fxml), UserChangePassword Scene (fxml), Userinformation Scene (fxml)	Hamza A. Kosa
27/5/2020	1.1	Admin Class, TableVariables Class, AdminInformationTable Scene (fxml)	Hamza A. Kosa & Hasan M. Al-Ani
28/5/2020	2.0	Ioperations Edited, Main Class Edit (Main), AdminAccess Class Edited, About Scene Edited (About), Main Login Page Edited (MainLoginPage), Admin Information Table	Hamza A. Kosa & Hasan M. Al-Ani
29/5/2020	3.0	UserPageController Edited, AdminPageController Edited, TableVariables Edited, LoginSystem Edited, Admin Class Edited, AdminLoginController Edited	Hamza A. Kosa & Hasan M. Al-Ani

Contents

1 Introduction

First, we would like to sincerely thank our beloved professor Taner Çevik who helped us during those hard times in the year 2020 and dedicated himself to teach and provide knowledge for his students during the world pandemic, we are truly thankful.

- **In our project we are looking for better- and high-quality banking system for all those who have access to our project, and it also provides an easy way to administrate the users registered, so users can do several things using this application from transferring money to deposit & withdraw and they sure want a trustworthy banking system to handle these serious operations**

1.1 Purpose

- Our top priority is our customer's complete trust in us and our application and of course fulfilling our customers need and we made our application based on these priorities, instead of wasting time going to the nearest bank or atm our customers can easily do everything they need from home using our easily understandable and usable application.

1.2 System Overview

- **in our project all types of people from welthy-companies owners to a simple college student are able to handle their bank accounts in the easiest yet the functional way, the user can transfer, deposit, and withdraw money, He or She can also check his/her account balance, change, and even view his/her account information, as well as for the administrator is able to add, remove a customer from the customers list provided in an individual window.**

1.3 Design Objectives

- What differs this application from other applications is the simplicity it provides, but don't let that simple look fool you my dear reader, as the known phrase says "don't judge a book by its cover".

- **In our project, we give the user many options to do what he wants and do many things such as sending to any place in the world, It also allows the user to withdraw some of his money, Or all of it, anything that works for our customers works for us as long as our customers are satisfied, The user can deposit money as far as he have some, changing the account password is also a provided option as well as viewing their account information.**

- And for the admin he can easily manage the customers registered in the application, the admin can see the list of all the bank's customers with names, ID's, passwords, and balances listed in a table organized way that provides an easy way for the admin to function, the admin is able to add a customer by simply typing a name and a password as well as for the dilatation of a customer the admin is asked to enter the ID of the customer he or she want to remove from the customers list.

1.4 References

- Of course, we won't forget to mention the education and knowledge that our beloved professor Taner Çevik gave us during this term despite the conditions and recent events, we would love to mention that without him we wouldn't be able to make any of this possible.

- https://harmash.com/home/ harmash is an Arabic website was used to provide help with design patterns.

- https://www.ziraatbank.com.tr/en ziraat bank website provided some information about the main functions any banking system should include.

- https://www.tutorialspoint.com/java8/index.htm tutorials point website provided knowledge used in methods and helped with the program coding.

- Various YouTube videos was relied on for different points such as how to use scene builder and helped us solving some problems we faced.

- Some help from also provided a good amount of help that we thank everyone for.

- https://www.lucidchart.com/pages/ lucidchart website helped us designing the UML

1.5 Definitions, Acronyms, and Abbreviations

- **Polymorphism:** is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object., Any Java object that can pass more than one IS-A test is considered to be polymorphic. In Java, all Java objects are polymorphic since any object will pass the IS-A test for their own type and for the class Object, it is important to know that the only possible way to access an object is through a reference variable. A reference variable can be of only one type. Once declared, the type of a reference variable cannot be changed.

- **Inheritance:** can be defined as the process where one class acquires the properties (methods and fields) of another. With the use of inheritance, the information is made manageable in a hierarchical order, the class which inherits the properties of other is known as subclass (derived class, child class) and the class whose properties are inherited is known as superclass (base class, parent class).

- **Overriding:** if a class inherits a method from its superclass, then there is a chance to override the method provided that it is not marked final, The benefit of overriding is: ability to define a behavior that's specific to the subclass type, which means a subclass can implement a parent class method based on its requirement, In object-oriented terms, overriding means to override the functionality of an existing method.

- **Abstraction:** is the quality of dealing with ideas rather than events. For example, when you consider the case of e-mail, complex details such as what happens as soon as you send an e-mail, the protocol your e-mail server uses are hidden from the user. Therefore, to send an e-mail you just need to type the content, mention the address of the receiver

1 Design Overview

2.1 Introduction

- We used JAVA FXML for our project for our design, because fxml is more flexible then the other types of java, in our application we believe that the design isn't important as the content, yet we tried to make it comfortable for the eye and as arranged as possible, so the customers can use it easily and feel comfortable while using the application, Also Scene Builder was used to facilitate the design work with fxml, we also used JavaCSS for buttons and text design, for the UML class diagram lucidchart website was used.

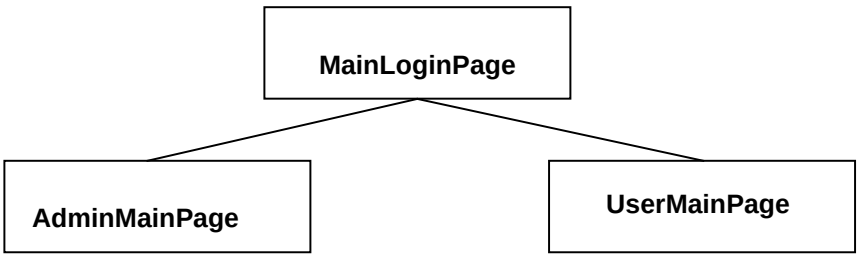
2.2 Environment Overview

- Our Project is made up of several scenes each scene requires the user to click on the provided buttons and labels to go to the next scene, for example for the admin to login it takes him to type a specific name and password that is registered as an admin in the admin file, the main admin page scene will open after typing the correct username and password, in the admin page scene there are specific buttons named after each function the button is assigned to, for example if the admin clicks on "Add User" the add user scene will open and show its contents.

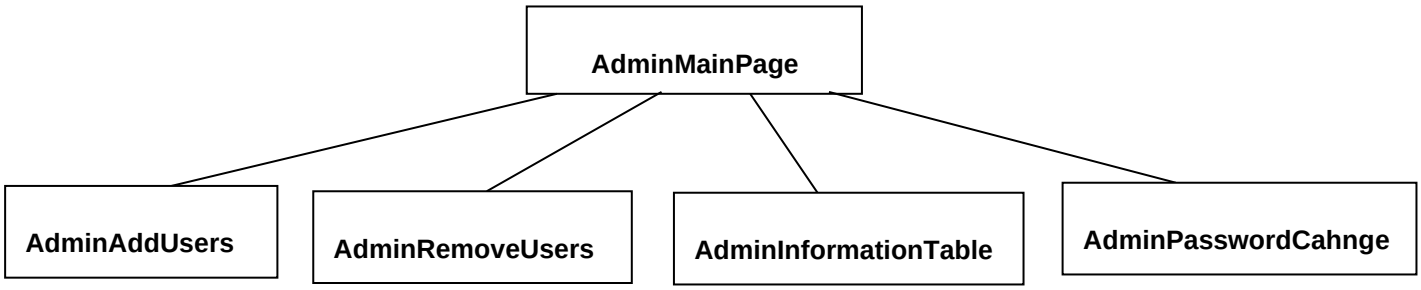
2.3 System Architecture

- Our application is based on three main scenes or pages, the mandatory and the 1st page on our application is the login page "MainLoginPage" in which asks the user to choose from 2 radio buttons whether he or she want to login as an admin or as a user, the admin choice goes to the admin main page "AdminMainPage" and it's one of the three main scenes, the user choice goes directly to the user's main page "UserMainPage" and it's the 3rd main scene in our application.

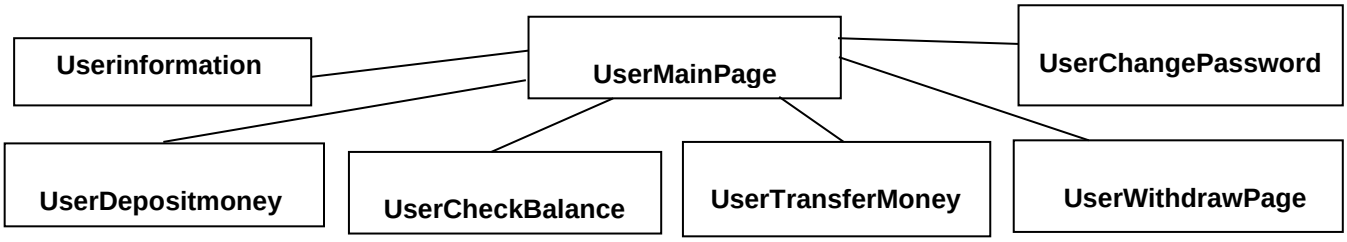
2.3.1 MainLoginPage Subsystem



2.3.2 AdminMainPage Subsystem



2.3.3 UserMainPage Subsystem



2.4 Constraints and Assumptions

Our Application is quite simple it doesn't have many constrains, these few things should be mentioned though,

1. *The admin's account's password cannot be changed.*
2. *A new admin account cannot be added, a bank administrator is considered permanent and not like the users' "customers" are always changing, a signup option for the admin isn't needed, even if such an option was available it would risk the application's security.*
3. *Costumer's maximum deposit, withdraw, or transfer is 999999, so customers can't deal with millions all at once.*
4. *The user cannot change his or her password to the same old password.*
5. *The user's / admin's ID cannot be changed, otherwise it's automatically generated when a new account is made.*

3 Interfaces & Data Stores

3.1 System Interfaces

3.1.1 Main Interface (MainLoginPage.fxml)

The main interface shows the login options in which the user of the application can choose whether to login as an admin or as a user, this interface gathers data for users and administrators from 2 files (admin.txt) and (user.txt) also this FXML scene is lunched using the class (Main), this interface also includes the (About) button, the buttons included inside this interface runs through the (MainController).

3.1.2 About (About.fxml)

This interface shows some information about the smart creators of this application all within some contact information and a brief paragraph about the application's purpose.

3.2 Admin Interface (AdminMainpage.fxml)

This interface is dedicated for the admin login page or the admin main page, this interface shows the admin his/her available options which will be mentioned, this interface is lunched through the (MainController), and runs through the (AdminLoginController).

3.2.1 Admin Add Users (AdminAddUsers.fxml)

This interface allows the admin to add users by typing the username and password then clicking on the (ADD) button, this scene was created using scene builder application, this interface is launched using the (AdminLoginController), and runs using the (AdminPageController), when the admin adds a new user the data is saved using the class (LoginSystem.signUp) in the file (Users.txt).

3.2.2 Admin Remove Users (AdminRemoveUsers.fxml)

This interface allows the admin to remove users by typing the username and password then clicking on the (REMOVE) button, this scene was created using scene builder application, this interface is launched using the (AdminLoginController), and runs using the (AdminPageController), when the admin removes a user the (AdminPageController.deleteUserButton) method runs the (AdminAccess.removeUser) method.

3.2.3 Admin View Users (AdminInformationTable.fxml)

This interface preview a table list with all the customers in the application when the admin clicks on (Load Information) all the customer's information will be shown in the table, this interface runs through (AdminLoginController), when the admin clicks on (Load Information Button) the (AdminLoginController.LoadTable) method runs (Admin.tab) to gather the users information from the file (Users.txt).

3.2.4 Admin Password Change (AdminPasswordChange.fxml)

This interface allows the admin to change any user's password in case the user forgot his password, this scene requires the user's ID, password, and the new password should be entered two times for verification, this interface is launched using the (AdminLoginController), and runs through the (AdminPageController), when the admin enters the user's information with the new password and click on the (Change) button the (AdminPageController.resetPasswordButton) method launches and changes the password from the file (Users.txt).

3.3 User Interface (UserMainPage.fxml)

This interface is the main page of the user's page, it shows the user's available options which will be mentioned further in the report, the user interface also lunches through the (MainController), and runs through the (UserLoginController).

3.3.1 User Password Change (UserChangePassword.fxml)

This interface allows the user to change his password by typing the old password then typing the new one and confirming it, this interface is launched using the (UserLoginController), and runs through the (UserPageController).

3.3.2 User Account Balance Check (UserCheckBalance.fxml)

This interface allows the user to check his/her account balance by opening the (Check Balance) button then hovering the mouse over the balance label, this interface is launched using the (UserLoginController), and runs through the (UserPageController).

3.3.3 User Money Deposit (UserDepositMoney.fxml)

This interface allows the user to deposit his/her money by clicking on the (Deposit Money) button then typing the requested money value the user wants to deposit, this interface is launched using the (UserLoginController), and runs through the (UserPageController).

3.3.4 User Account Information (Userinformation.fxml)

This interface allows the user to check on his/her account information by clicking on the (Information) button then clicking on (Load Information) button to view the user's information, this interface is launched using the (UserLoginController), and runs through the (UserPageController).

3.3.5 User Money Transfer (UserTransferMoney.fxml)

This interface allows the user to transfer his/her money by first clicking on the (Transfer Money) button then typing the money value requested for the transaction then pressing on the (Transfer) button, this interface is launched using the (UserLoginController), and runs through the (UserPageController).

3.3.6 User Money Withdraw (UserWithdrawPage.fxml)

This interface allows the user to withdraw his/her money, by clicking on the (Withdraw Money) button the money withdraw popup will open, the user needs to type the money value he/she need to withdraw, by clicking on (Withdraw) button the money value will be withdrawn and removed from the user balance, this interface is also launched using the (UserLoginController), and runs through the (UserPageController).

3.4 Data Stores

Only 4 files were used in this project

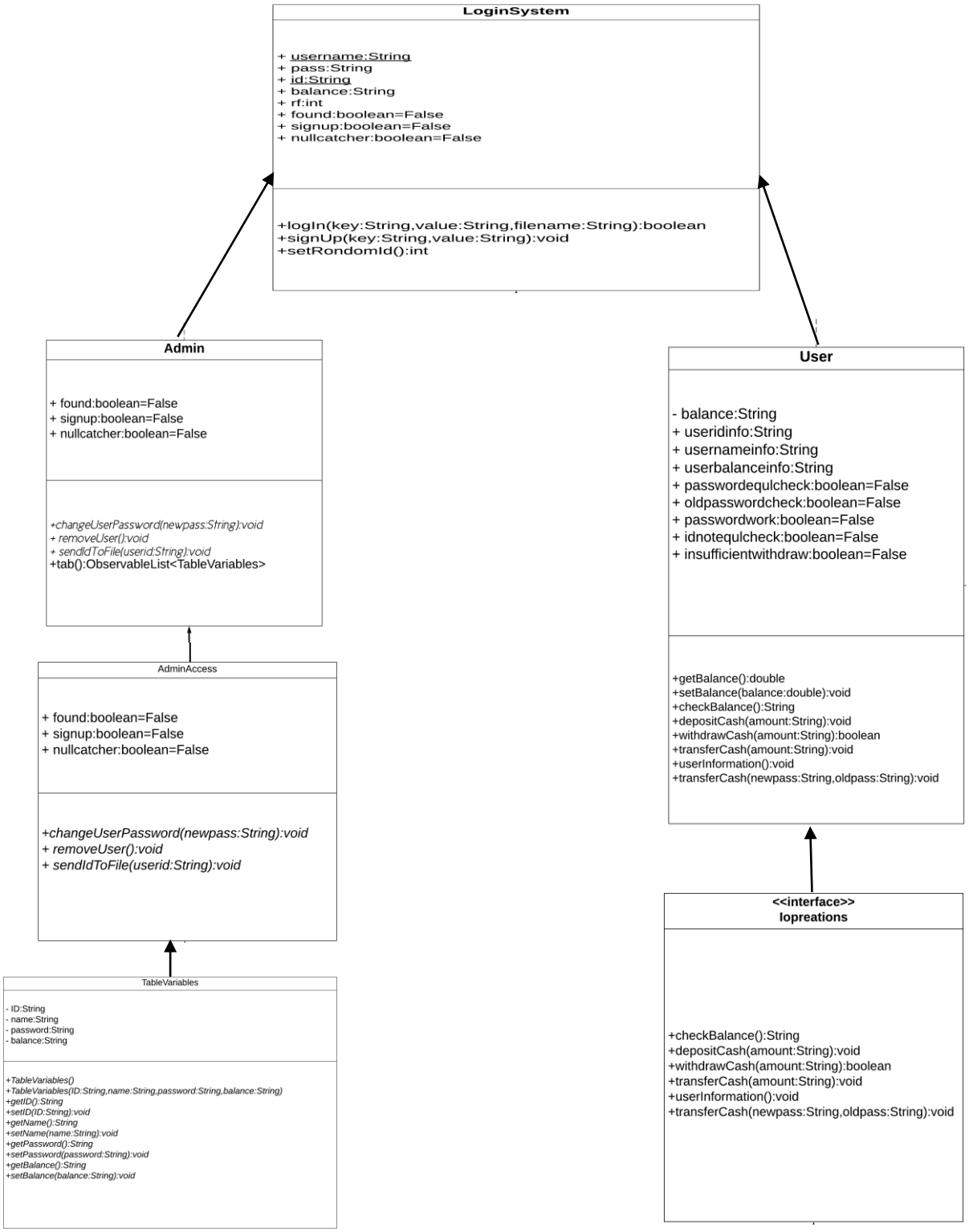
1. *Admins information file (Admins.txt) this file is used to store all admin's information such as ID, Name, Password.*
2. *Users information file (Users.txt) this file is used to store all user's information such as ID, Name, Password, and Balance.*
3. *Get ID (getId.txt) this file is a temporary file used to get the user ID and compare it to the user's ID information when the user is doing an operation as Money Withdraw or Money Deposit.*
4. *Send ID (sendId.txt) this file is a temporary file used to get the user ID and compare it to the user's ID information when the user is doing an operation such as Money Transfer or Money or Changing the user's password.*

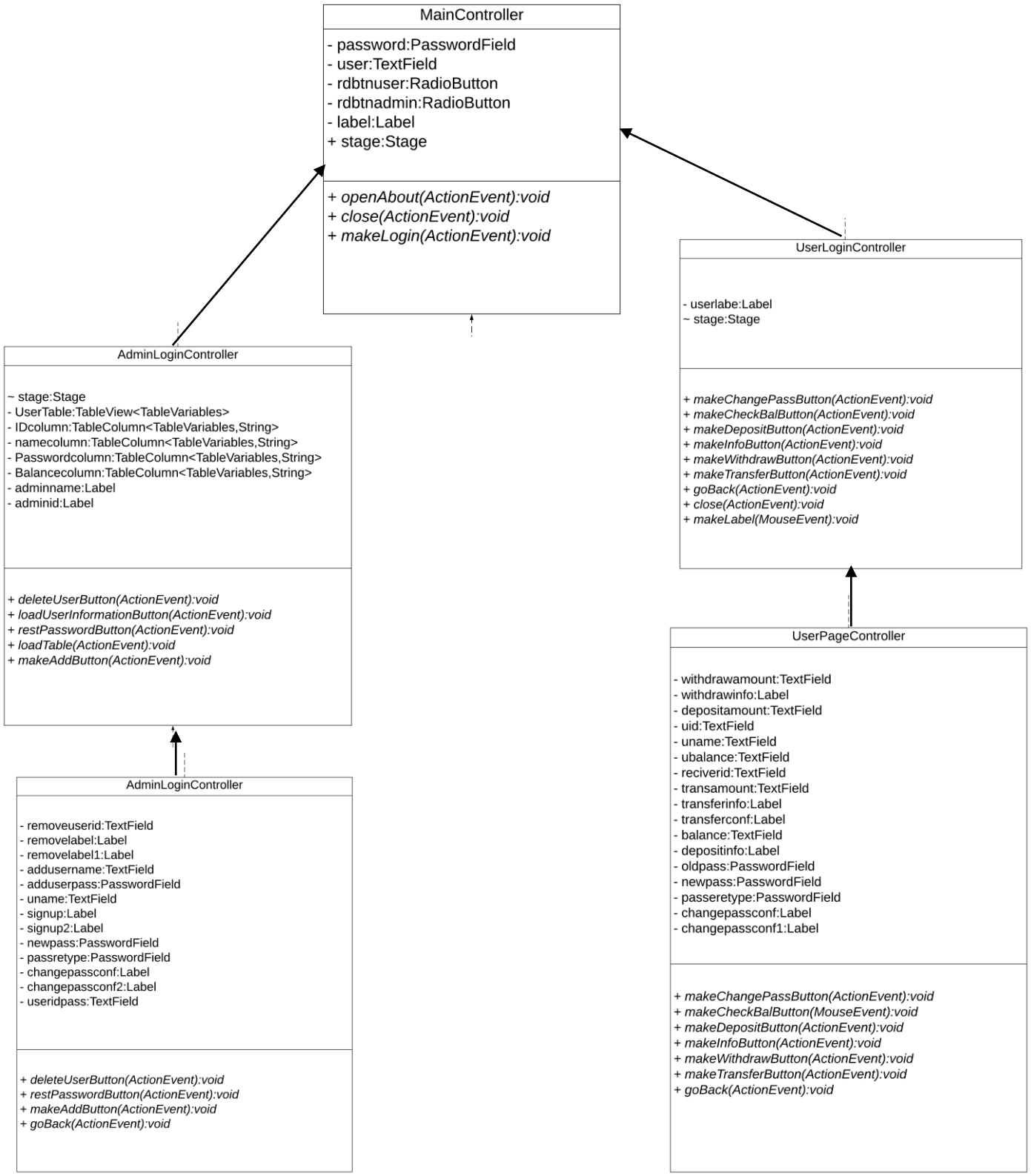
4 Structural Design

4.1 Design Discussion and Rationale

About our application design as said before we made as simple and efficient as possible, our application is mostly made of buttons and labels, two radio buttons are available for the user of the application to choose between the admin/user choice, the main page is as easy as it seems the user need to enter the name and password, also the user's and admin's main page is consistent of various buttons provided with icons and labels to declare each button's function.

4.2 Class Diagram




MainController
- password:PasswordField
- user:TextField
- rdbtnuser:RadioButton
- rdbtnadmin:RadioButton
- label:Label
+ stage:Stage
+ openAbout(ActionEvent):void
+ close(ActionEvent):void
+ makeLogin(ActionEvent):void
UserLoginController
- userlabe:Label
~ stage:Stage
+ makeChangePassButton(ActionEvent):void
+ makeCheckBalButton(ActionEvent):void
+ makeDepositButton(ActionEvent):void
+ makeInfoButton(ActionEvent):void
+ makeWithdrawButton(ActionEvent):void
+ makeTransferButton(ActionEvent):void
+ goBack(ActionEvent):void
+ close(ActionEvent):void
+ makeLabel(MouseEvent):void
AdminLoginController
~ stage:Stage
- UserTable:TableView<TableVariables>
- IDcolumn:TableColumn<TableVariables,String>
- namecolumn:TableColumn<TableVariables,String>
- Passwordcolumn:TableColumn<TableVariables,String>
- Balancecolumn:TableColumn<TableVariables,String>
- adminname:Label
- adminid:Label
+ deleteUserButton(ActionEvent):void
+ loadUserInformationButton(ActionEvent):void
+ restPasswordButton(ActionEvent):void
+ loadTable(ActionEvent):void
+ makeAddButton(ActionEvent):void
UserPageController
- withdrawamount:TextField
- withdrawinfo:Label
- depositamount:TextField
- uid:TextField
- uname:TextField
- ubalance:TextField
- reciverid:TextField
- transamount:TextField
- transferinfo:Label
- transferconf:Label
- balance:TextField
- depositinfo:Label
- oldpass:PasswordField
- newpass:PasswordField
- passeretype:PasswordField
- changepassconf:Label
- changepassconf1:Label
+ makeChangePassButton(ActionEvent):void
+ makeCheckBalButton(MouseEvent):void
+ makeDepositButton(ActionEvent):void
+ makeInfoButton(ActionEvent):void
+ makeWithdrawButton(ActionEvent):void
+ makeTransferButton(ActionEvent):void
+ goBack(ActionEvent):void
AdminLoginController
- removeuserid:TextField
- removelabel:Label
- removelabel1:Label
- addusername:TextField
- adduserpass:PasswordField
- uname:TextField
- signup:Label
- signup2:Label
- newpass:PasswordField
- passretype:PasswordField
- changepassconf:Label
- changepassconf2:Label
- useridpass:TextField
+ deleteUserButton(ActionEvent):void
+ restPasswordButton(ActionEvent):void
+ makeAddButton(ActionEvent):void
+ goBack(ActionEvent):void

4.3 Class Descriptions

4.4 Class: LoginSystem

- Purpose: To model the login system for the user and the admin.
- Constrains: The correct username and password should be entered.
- Persistent: Yes.

Attribute Descriptions

i. Attribute: username
Type: string
Description: stores the username
Constrains: shouldn't be left empty

ii. Attribute: pass
Type: string
Description: stores the password
Constrains: shouldn't be left empty

iii. Attribute: Id
Type: string
Description: stores the ID
Constrains: none (the ID is automatically generated)

iv. Attribute: balance
Type: string
Description: stores the balance
Constrains: none

v. Attribute: rf
Type: integer
Description: used to generate a random ID
Constrains: none

vi. Attribute: found
Type: boolean
Description: value changes to true if the user exists
Constrains: if the user doesn't exist it returns false

vii. Attribute: signup
Type: boolean
Description: checks if the username and password already exist
Constrains: none

viii. Attribute: nullcatcher
Type: boolean
Description: checks if the username and password entered are null
Constrains: none

Method Descriptions

i. Method: logIn
Return Type: boolean
Parameters: string key, string value, string filename
Return Value: true or false
Attributes read/used: Id, username, pass, balance, found
Methods called: none

ii. Method: setRandomId
Return Type: integer
Parameters: none
Return Value: random integer
Attributes read/used: rf
Methods called: none

iii. Method: singUp
Return Type: void
Parameters: string key, string value
Return Value: none
Attributes read/used: Id, username, pass, balance, signup, nullcatcher
Methods called: setRandomId

4.5 Class: Main

- Purpose: To start the application's primary stage.
- Constrains: none
- Persistent: Yes.

Attribute Descriptions – this class does not contain any attributes –

Method Descriptions

i. Method: start
Return Type: void
Parameters: Stage primaryStage
Return Value: none
Attributes read/used: none
Methods called: none

ii. Method: main
Return Type: void
Parameters: String [] args
Return Value: none
Attributes read/used: none
Methods called: none

4.6 Class: MainController

- Purpose: starts the admin, user pages, and about page, also contains the login system
- Constrains: none
- Persistent: Yes

Attribute Descriptions

i. Attribute: password
Type: PasswordField
Description: password field
Constrains: none

ii. Attribute: user
Type: TextField
Description: text field
Constrains: none

iii. Attribute: rdbtnuser

Type: RadioButton
Description: offers the user option in radio button
Constrains: none

iv. Attribute: rdbtnadmin
Type: string
Description: offers the admin option in radio button
Constrains: none

v. Attribute: label
Type: Label
Description: label
Constrains: none

vi. Attribute: stage
Type: Stage
Description: stage
Constrains: none

Method Descriptions

i. Method: openAbout
Return Type: void
Parameters: ActionEvent event
Return Value: none
Attributes read/used: stage
Methods called: none

ii. Method: close
Return Type: void
Parameters: ActionEvent event
Return Value: none
Attributes read/used: stage
Methods called: none

iii. Method: makeLogin
Return Type: void
Parameters: ActionEvent event
Return Value: none
Attributes read/used: username, user, pass, password, label
Methods called: logIn

4.7 Class: Admin

- Purpose: contains the needed methods for remove user, change user password, and the table list
- Constrains: runs only when the admin logs in
- Persistent: Yes

Attribute Descriptions

i. Attribute: passwordequlcheck
Type: boolean
Description: compares the old and new password
Constrains: none

ii. Attribute: idnotequlcheck
Type: boolean
Description: compares the old and new ID
Constrains: none

iii. Attribute: idnullcheck
Type: boolean
Description: checks if the id doesn't equal null
Constrains: none

Method Descriptions

i. Method: changeUserPassword
Return Type: void
Parameters: String newpass
Return Value: none
Attributes read/used: none
Methods called: none

ii. Method: sendId
Return Type: void
Parameters: String userid
Return Value: none
Attributes read/used: none
Methods called: none

iii. Method: removeUser
Return Type: void
Parameters: none
Return Value: none
Attributes read/used: none
Methods called: none

iv. Method: tab
Return Type: ObservableList<TableVariables
Parameters: none
Return Value: ObservableList<TableVariables
Attributes read/used: none
Methods called: none

4.7.1 Class: AdminAccess

- Purpose: overrides the admin class methods
- Constrains: none
- Persistent: Yes

Attribute Descriptions – this class does not contain any attributes –

Method Descriptions

i. Method: sendIdToFile
Return Type: void
Parameters: String userid
Return Value: none
Attributes read/used: none
Methods called: none

ii. Method: removeUser
Return Type: void
Parameters: none
Return Value: none
Attributes read/used: Id, username, pass, balance, idnullcheck
Methods called: none

iii. Method: changeUserPassword
Return Type: void
Parameters: String newpass
Return Value: none
Attributes read/used: Id, username, pass, balance, idnotequl, passwordequlcheck
Methods called: none

4.7.2 Class: AdminLoginController

- Purpose: handles the admin login methods
- Constrains: works only when the admin logs in
- Persistent: Yes

Attribute Descriptions

i. Attribute: stage
Type: Stage
Description: used to close stage popup without stacking
Constrains: none

ii. Attribute: UserTable
Type: TableVariables
Description: contains users' information
Constrains: none

iii. Attribute: IDcolumn
Type: TableVariables, String
Description: contains the ID column in the users' information table
Constrains: none

iv. Attribute: namecolumn
Type: TableVariables, String
Description: contains the name column in the users' information table
Constrains: none

v. Attribute: Passwordcolumn
Type: TableVariables, String
Description: contains the password column in the users' information table
Constrains: none

vi. Attribute: Balancecolumn
Type: TableVariables, String
Description: contains the balance columns in the users' information table

vii. Attribute: adminname
Type: Label
Description: stores the admin name to show it on the main admin page scene
Constrains: none

viii. Attribute: adminid
Type: Label
Description: stores the admin name to show it on the main admin page scene
Constrains: none

Method Descriptions

i. Method: DeleteUserButton
Return Type: void
Parameters: ActionEvent event
Return Value: none
Attributes read/used: stage
Methods called: none

ii. Method: LoadUserInformation
Return Type: void
Parameters: ActionEvent event
Return Value: none
Attributes read/used: stage
Methods called: none

iii. Method: ResetPasswordButton
Return Type: void
Parameters: ActionEvent event
Return Value: none
Attributes read/used: stage
Methods called: none

iv. Method: LoadTable
Return Type: void
Parameters: ActionEvent event
Return Value: none
Attributes read/used: IDcolumn, namecolumn, passwordcolumn, Balancecolumn, UserTable
Methods called: none

v. Method: makeAddButton
Return Type: void
Parameters: ActionEvent event
Return Value: none
Attributes read/used: stage
Methods called: none

vi. Method: goBackAdmin
Return Type: void
Parameters: ActionEvent event
Return Value: none
Attributes read/used: none
Methods called: none

vii. Method: goBack
Return Type: void
Parameters: ActionEvent event
Return Value: none
Attributes read/used: none
Methods called: none

viii. Method: makeLabel
Return Type: void
Parameters: MouseEvent event
Return Value: none
Attributes read/used: adminname, usernameinfo, adminid, useridinfo
Methods called: none

4.7.3 Class: AdminPageController

- Purpose: contains action handlings for the admin page
- Constrains: used only within the admin page
- Persistent: Yes, within the admin page

Attribute Descriptions

i. Attribute: removeuserid
Type: TextField
Description: text field
Constrains: none

ii. Attribute: removelabel
Type: Label
Description: label
Constrains: none

iii. Attribute: removelabel1
Type: Label
Description: label
Constrains: none

iv. Attribute: addusername
Type: TextField
Description: text field
Constrains: none

v. Attribute: adduserpass
Type: PasswordField
Description: password field
Constrains: none

vi. Attribute: signuplabel
Type: Label
Description: label
Constrains: none

vii. Attribute: signuplabel2
Type: Label
Description: label
Constrains: none

viii. Attribute: newpass
Type: PasswordField
Description: password field
Constrains: none

ix. Attribute: passretype
Type: PasswordField
Description: password field
Constrains: none

x. Attribute: changepassconf
Type: Label
Description: label
Constrains: none

xi. Attribute: changepassconf1
Type: Label
Description: label
Constrains: none

xii. Attribute: useridpass
Type: TextField
Description: text field
Constrains: none

Method Descriptions

i. Method: deleteUserButton
Return Type: void
Parameters: ActionEvent event
Return Value: none
Attributes read/used: removeuserid, removelabel1, removelabel, idnullcheck,
Methods called: sendIdToFile, removeUser

ii. Method: resetPasswordButton
Return Type: void
Parameters: ActionEvent event
Return Value: none
Attributes read/used: passretype, newpass, changepassconf1, changepassconf, useridpass, passwordequlcheck, idnotequlcheck
Methods called: sendIdToFile, ChangeUserPassword

iii. Method: makeAddButton
Return Type: void
Parameters: ActionEvent event
Return Value: none
Attributes read/used: username, addusername, pass, adduserpass, nullcatcher, signuplabel, signuplabel2,
Methods called: signUp

iv. Method: goBack
Return Type: void
Parameters: ActionEvent event
Return Value: none
Attributes read/used: stage
Methods called: none

4.7.4 Class: TableVariables

- Purpose: runs the table information list for the admin view users' option
- Constrains: none
- Persistent: No

Attribute Descriptions

i. Attribute: ID
Type: String
Description: used to view the user's ID
Constrains: none

ii. Attribute: name
Type: String
Description: used to view the user's name
Constrains: none

iii. Attribute: password
Type: String
Description: used to view the user's password
Constrains: none

iv. Attribute: Balance
Type: String
Description: used to view the user's balance
Constrains: none

Method Descriptions

i. Method: getID
Return Type: String
Parameters: none
Return Value: none
Attributes read/used: ID
Methods called: none

ii. Method: setID
Return Type: void
Parameters: String ID
Return Value: none
Attributes read/used: ID
Methods called: none

iii. Method: getName
Return Type: String
Parameters: none
Return Value: name
Attributes read/used: name
Methods called: none

iv. Method: setName
Return Type: void
Parameters: String name
Return Value: none
Attributes read/used: name
Methods called: none

v. Method: getPassword
Return Type: String
Parameters: none
Return Value: none
Attributes read/used: password
Methods called: none

vi. Method: setPassword
Return Type: void
Parameters: String password
Return Value: none
Attributes read/used: password
Methods called: none

vii. Method: getBalance
Return Type: String
Parameters: none
Return Value: Balance
Attributes read/used: Balance
Methods called: none

viii. Method: setBalance
Return Type: void
Parameters: String Balance
Return Value: none
Attributes read/used: Balance
Methods called: none

4.8 Class: User

- Purpose: contains the needed methods for the user page
- Constrains: used while loged in as a user only
- Persistent: Yes, among the user classes

Attribute Descriptions

i. Attribute: Balance
Type: String
Description: contains the user's balance
Constrains: contains numbers from 0 to 9, balance cannot be negative

ii. Attribute: useridinfo
Type: String
Description: contains the user's ID
Constrains: contains only numbers, automatically generated

iii. Attribute: usernameinfo
Type: String
Description: contains the user's name
Constrains: none

iv. Attribute: userbalanceinfo
Type: String
Description: contains the user's balance
Constrains: none

v. Attribute: passwordequlcheck
Type: boolean
Description: contains the value true or false
Constrains: none

vi. Attribute: oldpasswordcheck
Type: boolean
Description: contains the value true or false
Constrains: none

vii. Attribute: passwordwork
Type: boolean
Description: contains the value true or false
Constrains: none

viii. Attribute: idnotequlcheck
Type: boolean
Description: contains the value true or false
Constrains: none

ix. Attribute: insufficientwithdraw
Type: boolean
Description: contains the value true or false
Constrains: none

Method Descriptions

i. Method: getBalance
Return Type: String
Parameters: none
Return Value: returns checkbalance method
Attributes read/used: none
Methods called: chechbalance

ii. Method: checkBalance
Return Type: String
Parameters: String balance
Return Value: returns fee
Attributes read/used: Id, username, pass, balance, writer
Methods called: none

iii. Method: depositCash
Return Type: void
Parameters: String amount
Return Value: none
Attributes read/used: Id, username, pass, balance, writer
Methods called: none

iv. Method: withdrawCash
Return Type: boolean
Parameters: String amount
Return Value: returns true or false
Attributes read/used: Id, username, pass, balance, writer, insufficientwithdraw
Methods called: none

v. Method: passwordChange
Return Type: void
Parameters: String newpass, String oldpass
Return Value: none
Attributes read/used: Id, username, pass, balance, writer, oldpasswordcheck, passwordequlcheck, passwordwork
Methods called:

vi. Method: transferCash
Return Type: void
Parameters: String amount
Return Value: none
Attributes read/used: Id, username, pass, balance, writer, idnotequlcheck, useridinfo
Methods called: none

vii. Method: userInformation
Return Type: void
Parameters: none
Return Value: none
Attributes read/used: Id, username, pass, balance, useridinfo, usernameinfo, userbalanceinfo
Methods called: none

4.8.1 Class: UserLoginController

- Purpose: handles the user login functions
- Constrains: only used within the user's interfaces and functuality
- Persistent: Yes, among the user's classes and interfaces

Attribute Descriptions

i. Attribute: userlabel
Type: Label
Description: label
Constrains: none

ii. Attribute: stage
Type: Stage
Description: used to close stage popup without stacking
Constrains: none

Method Descriptions

i. Method: makeChangePassButton
Return Type: void
Parameters: ActionEvent event
Return Value: none
Attributes read/used: stage
Methods called: none

ii. Method: makeCheckBalButton
Return Type: void
Parameters: ActionEvent event
Return Value: none
Attributes read/used: stage
Methods called: none

iii. Method: makeDepositButton
Return Type: void
Parameters: ActionEvent event
Return Value: none
Attributes read/used: stage
Methods called: none

iv. Method: makeInfoButton
Return Type: void
Parameters: ActionEvent event
Return Value: none
Attributes read/used: stage
Methods called: none

v. Method: makeWithdrawButton
Return Type: void
Parameters: ActionEvent event
Return Value: none
Attributes read/used: stage
Methods called: none

vi. Method: makeTransferButton
Return Type: void
Parameters: ActionEvent event
Return Value: none
Attributes read/used: stage
Methods called: none

vii. Method: makeTransferButton
Return Type: void
Parameters: ActionEvent event
Return Value: none
Attributes read/used: stage
Methods called: none

viii. Method: goBack
Return Type: void
Parameters: ActionEvent event
Return Value: none
Attributes read/used: none
Methods called: none

ix. Method: close
Return Type: void
Parameters: ActionEvent event
Return Value: none
Attributes read/used: stage
Methods called: none

4.8.2 Class: UserPageController

- Purpose: handles the user login methods
- Constrains: used among user methods only
- Persistent: Yes, among user's classes and functions

Attribute Descriptions

i. Attribute: withdrawamount
Type: TextField
Description: text field for the among of money the user needs to withdraw
Constrains: money amount maximum limit is 999999

ii. Attribute: trnasamount
Type: TextField
Description: text field for the amount of money the user wants to transfer
Constrains: maximum amount of money for transfer is 999999

iii. Attribute: oldpass
Type: PasswordField
Description: password field for the old password
Constrains: none

Method Descriptions

i. Method: makeChangePassButton
Return Type: void
Parameters: ActionEvent event
Return Value: none
Attributes read/used: passretype, oldpass, newpass, changepassconf, changepassconf1, oldpasswordcheck
Methods called: passwordChange

ii. Method: makeTransferButton
Return Type: void
Parameters: ActionEvent event
Return Value: none
Attributes read/used: receiverid, trnasamount, transferinfo, transferconf
Methods called: none

iii. Method: makeInfoButton
Return Type: void
Parameters: ActionEvent event
Return Value: none
Attributes read/used: depositamount, depositinfo
Methods called: none

5 Supplementary Documentation

- CSS files for most hovers and mouse click
- *SceneBiulder for the GUI design and functionality*
- *Javafx SDK 11.0.2 was also used to help us with the application*